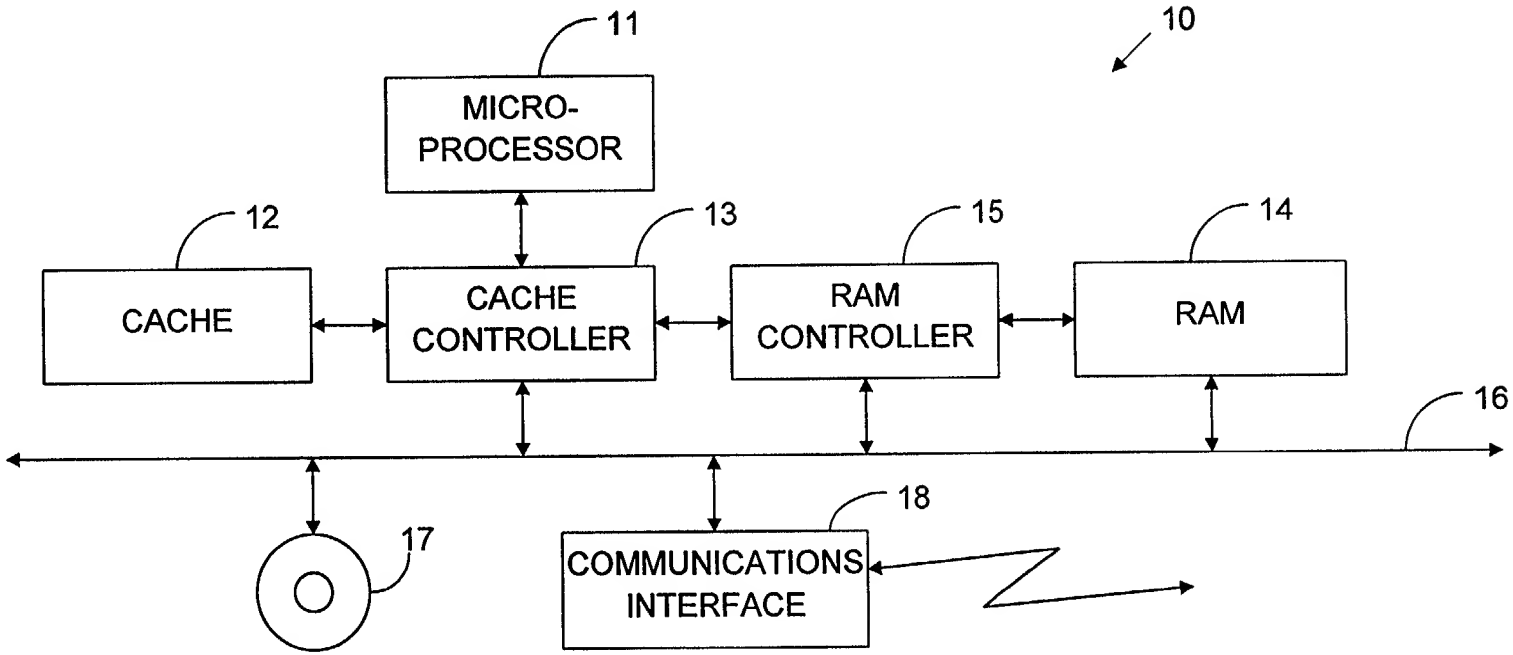
10
11
MICRO-
PROCESSOR
12
CACHE
13
CACHE
CONTROLLER
15
RAM
CONTROLLER
14
RAM
16
17
18
COMMUNICATIONS
INTERFACE

FIG. 1

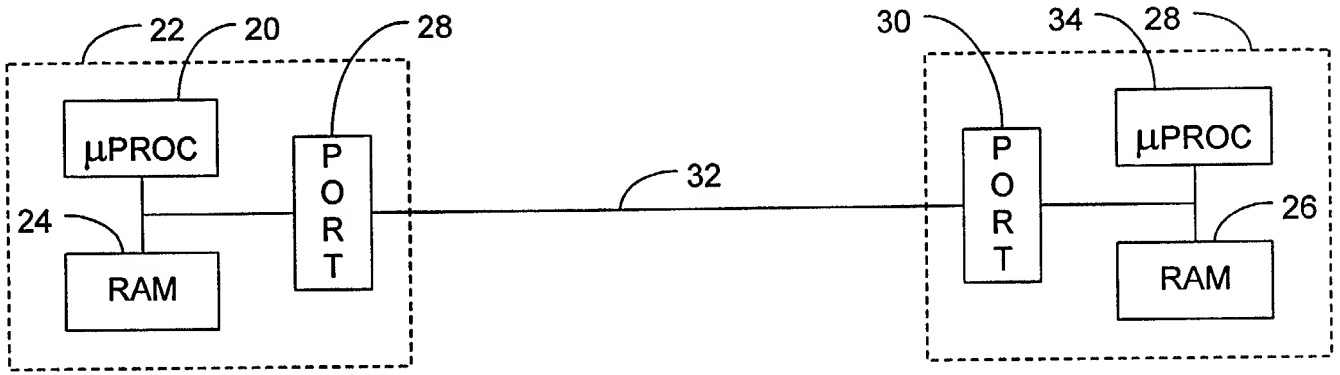

FIG. 2

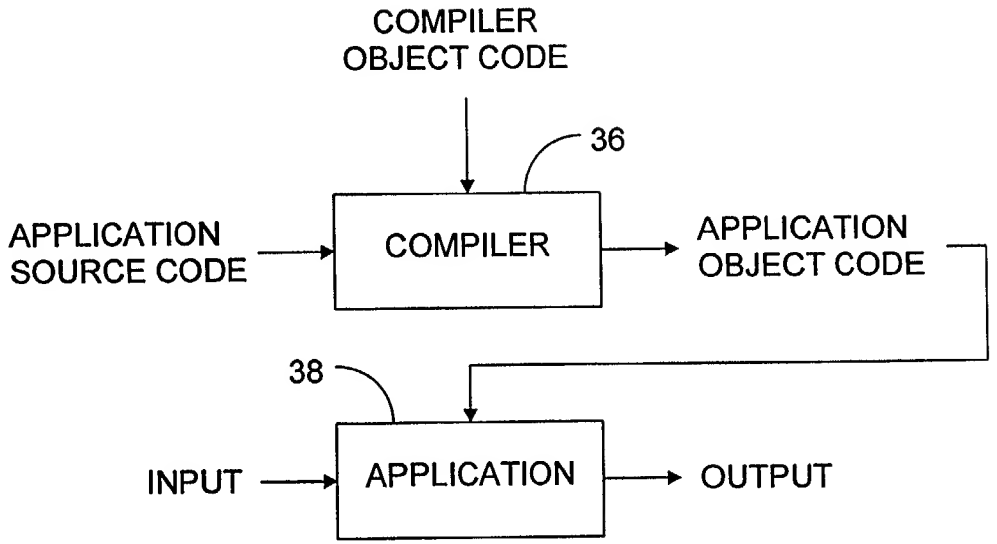

FIG. 3

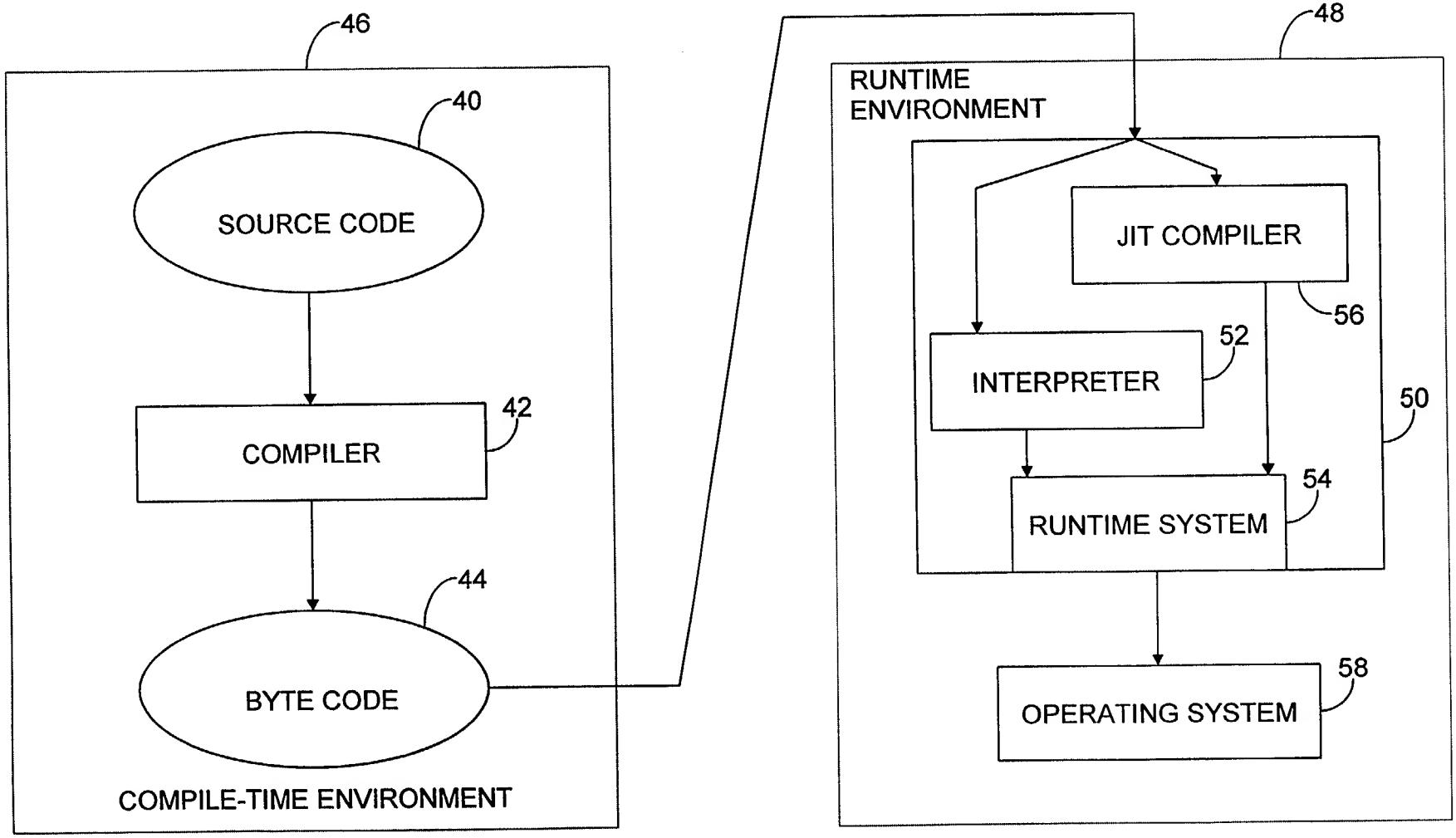
46
40
SOURCE CODE
COMPILER
42
44
BYTE CODE
COMPILE-TIME ENVIRONMENT
48
RUNTIME
ENVIRONMENT
JIT COMPILER
56
52
INTERPRETER
50
54
RUNTIME SYSTEM
58
OPERATING SYSTEM

FIG. 4

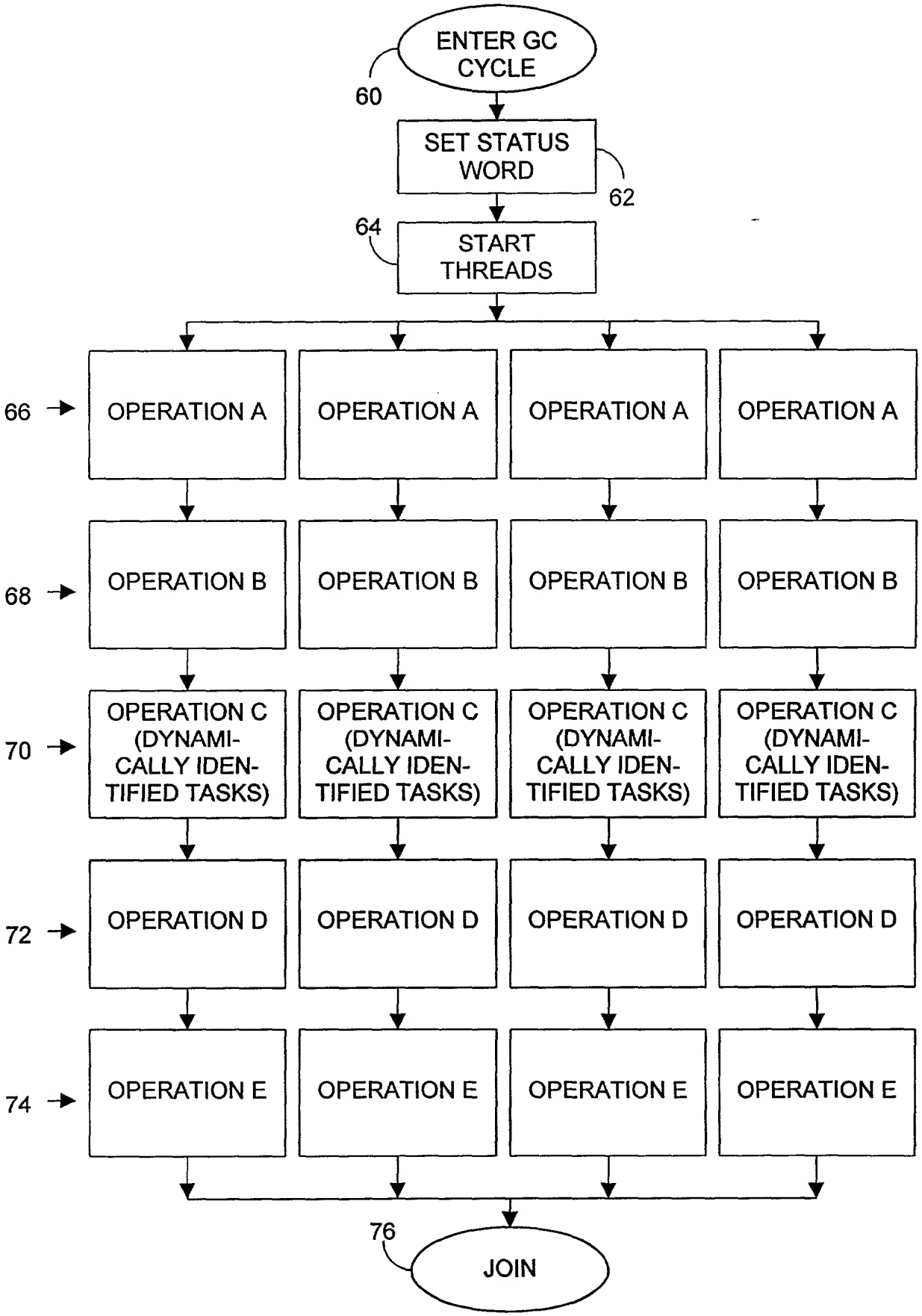
ENTER GC CYCLE
60
SET STATUS WORD
62
64
START THREADS
66
OPERATION A
OPERATION A
OPERATION A
OPERATION A
68
OPERATION B
OPERATION B
OPERATION B
OPERATION B
70
OPERATION C (DYNAMI-CALLY IDEN-TIFIED TASKS)
OPERATION C (DYNAMI-CALLY IDEN-TIFIED TASKS)
OPERATION C (DYNAMI-CALLY IDEN-TIFIED TASKS)
OPERATION C (DYNAMI-CALLY IDEN-TIFIED TASKS)
72
OPERATION D
OPERATION D
OPERATION D
OPERATION D
74
OPERATION E
OPERATION E
OPERATION E
OPERATION E
76
JOIN

FIG. 5

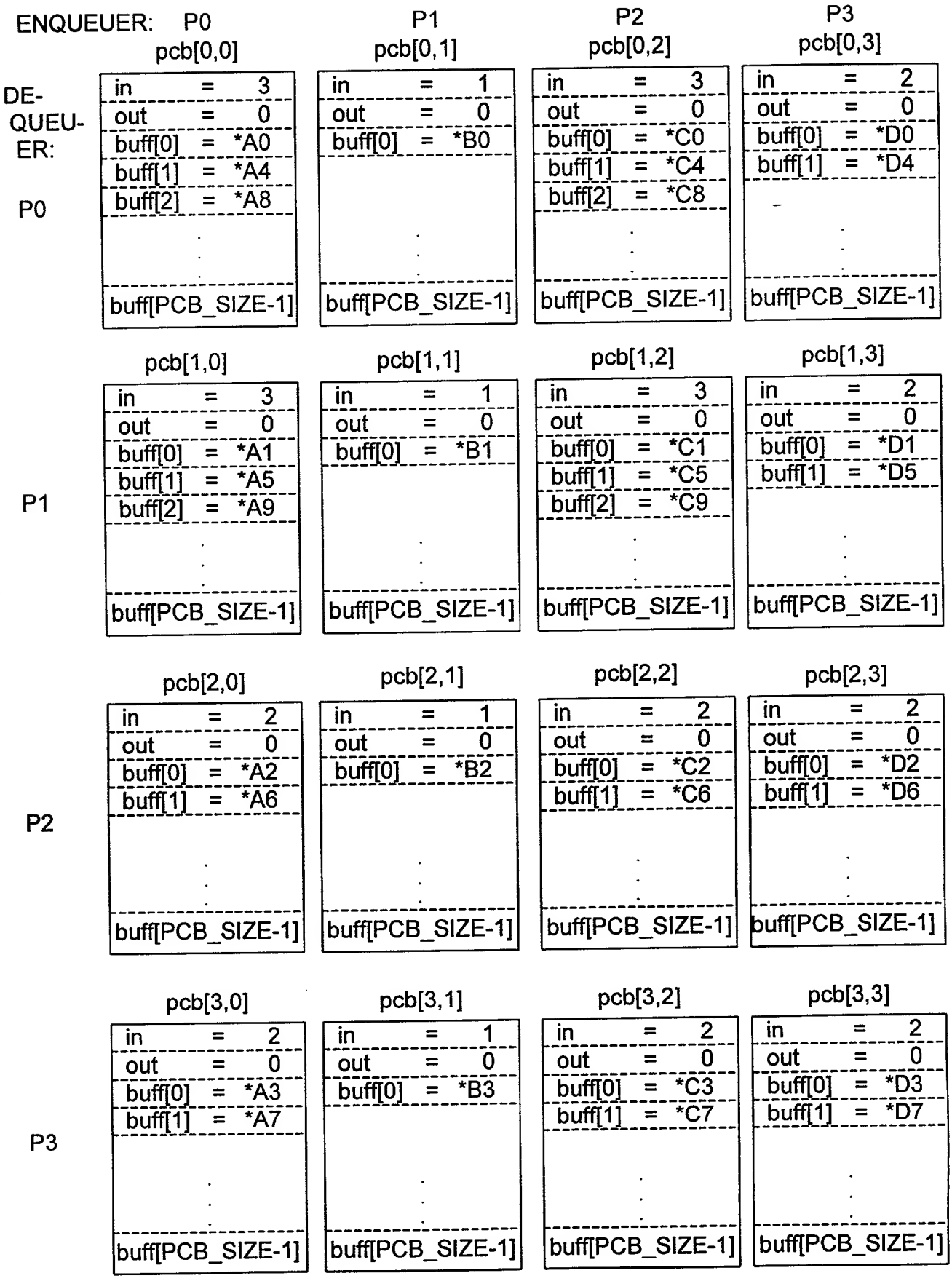
ENQUEUER: P0
P1
P2
P3
DE-QUEU-ER:
P0
P1
P2
P3
pcb[0,0]
in = 3
out = 0
buff[0] = *A0
buff[1] = *A4
buff[2] = *A8
buff[PCB_SIZE-1]
pcb[0,1]
in = 1
out = 0
buff[0] = *B0
buff[PCB_SIZE-1]
pcb[0,2]
in = 3
out = 0
buff[0] = *C0
buff[1] = *C4
buff[2] = *C8
buff[PCB_SIZE-1]
pcb[0,3]
in = 2
out = 0
buff[0] = *D0
buff[1] = *D4
buff[PCB_SIZE-1]
pcb[1,0]
in = 3
out = 0
buff[0] = *A1
buff[1] = *A5
buff[2] = *A9
buff[PCB_SIZE-1]
pcb[1,1]
in = 1
out = 0
buff[0] = *B1
buff[PCB_SIZE-1]
pcb[1,2]
in = 3
out = 0
buff[0] = *C1
buff[1] = *C5
buff[2] = *C9
buff[PCB_SIZE-1]
pcb[1,3]
in = 2
out = 0
buff[0] = *D1
buff[1] = *D5
buff[PCB_SIZE-1]
pcb[2,0]
in = 2
out = 0
buff[0] = *A2
buff[1] = *A6
buff[PCB_SIZE-1]
pcb[2,1]
in = 1
out = 0
buff[0] = *B2
buff[PCB_SIZE-1]
pcb[2,2]
in = 2
out = 0
buff[0] = *C2
buff[1] = *C6
buff[PCB_SIZE-1]
pcb[2,3]
in = 2
out = 0
buff[0] = *D2
buff[1] = *D6
buff[PCB_SIZE-1]
pcb[3,0]
in = 2
out = 0
buff[0] = *A3
buff[1] = *A7
buff[PCB_SIZE-1]
pcb[3,1]
in = 1
out = 0
buff[0] = *B3
buff[PCB_SIZE-1]
pcb[3,2]
in = 2
out = 0
buff[0] = *C3
buff[1] = *C7
buff[PCB_SIZE-1]
pcb[3,3]
in = 2
out = 0
buff[0] = *D3
buff[1] = *D7
buff[PCB_SIZE-1]
80

FIG. 6

```
1   static void enque(ParallelThread *pt, java_lang_Object *v) {
2           int p = pt->number; /* process id */
3           sq_nexts *local = (sq_nexts *) pt->data1; /* local is local to thread p */
4           while (PCBfull(Q->pcb[local->nextpush,p])) {
5                   local->nextpush = mod(local->nextpush + 1, n)
6           }
7           PCBpush(v,Q->pcb[local->nextpush,p]); /* push item */
8           local->nextpush = mod(local->nextpush + 1, n);
9   }

1   typedef struct {
2           int in; /* counter of total number of elements inserted into PCB */
3           int out; /* counter of total number of elements deleted from the PCB */
4           java_lang_Object *buff[PCB_SIZE]; /* the PCB buffer, which is ``circular''*/
5   } PCB;

1   static bool_t PCBfull(PCB *b) {
2           if (mod(b->in - b->out, PCB_SIZE) == PCB_SIZE - 1) /* leave empty space */
3           return TRUE;
4           else return FALSE;
5   }

1   static PCBpush(java_lang_Object *v, PCB *b) {
2           b->buff[b->in] = v;
3           b->in = mod(b-> in + 1, PCB_SIZE);
4   }

1   static int mod(int x, int n){
2           while(x >= n) x = x - n;
3           while(x < 0) x = x + n;
4           return x;
5   }
```

Fig. 7

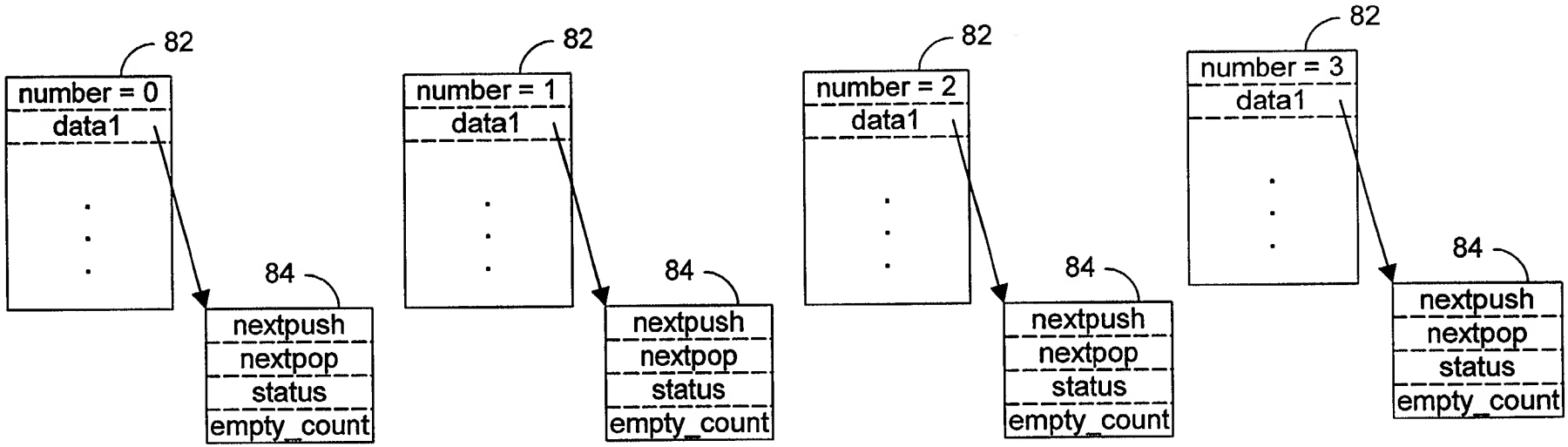
82
number = 0
data1
84
nextpush
nextpop
status
empty_count
82
number = 1
data1
84
nextpush
nextpop
status
empty_count
82
number = 2
data1
84
nextpush
nextpop
status
empty_count
82
number = 3
data1
84
nextpush
nextpop
status
empty_count

FIG. 8

```
1   static java_lang_Object *deque(ParallelThread *pt){
2           int p = pt->number; /* process id */
3           sq_nexts *local = (sq_nexts *) pt->data1; /* local is local to thread p */
4           int j;
5           const term_limit = 2*n; /* limit set after which termination is attempted */
6           while (PCBempty(Q->pcb[p,local->nextpop])) {
7                   local->nextpop = mod(local->nextpop + 1,n);
8                   local->empty_count = local->empty_count+1;
9                   if (local->empty_count == term_limit) {
10                          local->status = inactive;
11                          for(j=0; j=n-1; j++) {
12                                  if (! PCBempty(Q->pcb[j,p]))
13                                          local->status = active;
14                          }
15                          if (local->status == inactive)
16                                  mark_self_inactive(p,&statusBitmap);
17                          if (!StatusBitmap) /* all threads are inactive, return */
18                                  return NULL; /* system termination state reached */
19                          else local->empty_count = 0;
20                  }
21          }
22          if (local->status == inactive){
23                  local->status = active;
24                  mark_self_active(p,&statusBitmap);
25          }
26          int pop = local->nextpop;
27          local->nextpop = mod(local->nextpop + 1, n);
28          return PCBpop(Q->pcb[p,pop]);
29  }

1   static bool_t PCBempty(PCB *b) {
2           return (b->in == b->out);
3   }

1   static java_lang_Object *PCBpop(PCB *b) {
2           java_lang_Object *v;
3           v = b->buff[b->out];
4           b->out = mod(b->out +1, PCB_SIZE);
5           return v;
6   }
```

Fig. 9

```
1   static void mark_self_inactive(int self, int *pStatusBitmap) {
2           int oldValue,newValue;
3           do {
4                   oldValue = *pStatusBitmap;
5                   newValue = oldValue & (~(1<<self));
6                   newValue = casInt(newValue, oldValue, pStatusBitmap);
7           } while (newValue != oldValue) ;
8   }
```

```
1   static void mark_self_active(int self, int *pStatusBitmap) {
2           int oldValue,newValue;
3           do {
4                   oldValue = *pStatusBitmap;
5                   newValue = oldValue | (1<<self);
6                   newValue = casInt(newValue, oldValue, pStatusBitmap);
7           } while (newValue != oldValue) ;
8   }
```

Fig. 10